Assignment 2 Guide

This guide is intended to help students to get started on the assignment.

Here are the recommended steps:

1. Build the University window, skipping anything that's difficult
2. Build the Add Student window, skipping anything that's difficult
3. Build the Student window, skipping anything that's difficult
4. Complete the difficult bits

You **MUST** also fill in the progress.txt file and submit to PLATE in regular increments otherwise you will receive mark penalties.

First pass: 68%

In the first pass, you are advised to skip over the difficult bits so that you can get a higher mark more quickly. What is suggested below is a guide. You may skip over even more, and come back to it later. That is up to you. For example, in the first pass, it is suggested below to do CSS styling and layout, but it is perfectly reasonable to skip over that and come back to it later.

Passing the recommended verification points below will give you 56 potential marks on PLATE. Assuming you have also used the correct package structure, applied the MVC architecture with FXML correctly, used the correct JavaFX property patterns, and stored the attendance mode as "ft" or "pt" via userData, then you should also receive another 12 marks after the due date giving a total of 68 (CREDIT).

Below are the recommended verification points for your first pass. After this list is a step-by-step development guide.

Launch the application
[y] The UTS Logo is shown
[y] The "Timetable System" heading is shown
[y] "No students" is shown in the empty list
[y] "Add Student" appears and is enabled
[y] "Remove Student" appears and is disabled
[y] "Login" appears and is disabled
[y] The layout arranges the header, content and footer sections vertically
[y] All colours and fonts are correct
[y] The spacing and padding look like the screenshots
Click "Add Student"
[y] The "Add Student" window is shown
[y] The "Add new student" heading is shown
[y] The student number label and text field are shown
[y] The student name label and text field are shown
[y] The attendance label, "Full Time" radio button and "Part Time" radio button are shown
[y] The scholarship label and checkbox are shown

[y] The "Cancel" button is shown and enabled
[?] **The "Add" button is shown** and disabled
[y] The layout arranges the header, content and footer sections vertically
[y] The form is laid out in a GridPane
[y] All colours and fonts are correct including the error message
[y] The error message is included in the GridPane, centred horizontally spanning 2 columns
[?] The spacing and padding look like the screenshots
Click "Cancel"
[y] The "Add Student" window is closed
Click "Add Student" again
Enter student number 12345678
[?] The "Add" button remains disabled
Enter student name Bianca Sladen
[?] The "Add" button remains disabled
Select "Full Time"
[?] The "Add" button becomes enabled
Select the "Scholarship" checkbox
Delete the student number and the re-enter it
[?] The "Add" button becomes disabled and then re-enabled
Delete the student name and then re-enter it
[?] The "Add" button becomes disabled and then re-enabled
Click the "Add" button
[y] The "Add Student" window is closed
[y] "12345678 - Bianca Sladen" is shown in the student list in the university window
Click the "Add Student" button
Enter details "12345678", "Hugo Aitken", "Full Time" and no scholarship
Click "Add"
[?] The window does not close and the error "Student already exists" is shown
Change the student number to "49287512" and click "Add"
[y] The "Add Student" window is closed and "49287512 - Hugo Aitken" is immediately added to the student list in the university window
Add a student "23232323", "Jessica Sneddon", "Part Time", no scholarship
Add a student "11111111", "Dakota Cavill", "Full Time", scholarship
[y] These two students are immediately added to the student list in the university window
Select Bianca Sladen from the list
[y] At this moment, the "Remove Student" and "Login" buttons become enabled
Click "Login"
[y] The "Student" window is shown
[y] The "Logged in as Bianca Sladen" heading is shown
[y] The student number label and text (12345678) is shown
[?] The student attendance label and text ("Full Time") is shown
[?] The student scholarship label and text ("Yes") is shown
[?] **The "Withdraw" and "Enrol" buttons are both shown** and disabled. **The "Logout" button is shown** and enabled.
[y] The "My Activities" table shows column headings: Subject, Group, Activity, Day, Start, Duration, Room, Capacity, Enrolled.
[y] The "My Activities" table is empty and shows "Not enrolled in any activities"
[y] The subjects ComboBox is shown and it displays the subjects from the model (i.e. "48024 Applications Programming" and "31284 Web Services Development")
[y] The subject activities table is empty and shows column headings: Subject, Group, Activity, Day, Start, Duration, Room, Capacity, Enrolled.

[y] The layout arranges the header, content and footer sections vertically
[y] The "Logged in as ..." heading is in the top left, and the student details GridPane is in the top right
[y] The "My Activities" heading is aligned left, and the "Withdraw" button is aligned right
[y] The "Enrol into subject" heading and ComboBox are aligned left, and the "Enrol" button is aligned right
[y] All colours and fonts are correct
[?] The spacing and padding look like the screenshots
Select "48024 Applications Programming" from the subjects ComboBox
[?] The Subject column shows the correct subject number (and ONLY the subject number) of each activity in the selected subject
[y] The Group, Activity, Day, Start, Duration, Room, Capacity and Enrolled columns show the corresponding properties from the Activity model data for each activity in the subject
Select Lec1 activity 1 from the subject activities table
[?] The "Enrol" button becomes enabled
Click "Enrol" and make sure Lec1 activity 1 is still selected in the table
[?] The "Enrol" button becomes disabled
[y] The selected activity is immediately added to the student's activities and shows in the "My Activities" table
[y] The "Enrolled" column for the selected activity increases to 1 in both tables
Select Lec1 activity 1 from the "My Activities" table
[?] The "Withdraw" button becomes enabled
Click "Withdraw"
[y] The selected activity is removed from the "My Activities" table
[y] The "enrolled" column for that same activity is decreased to 0 in the subject activities table.
[?] The "Withdraw" button becomes disabled
Select Lec1 activity 1 from the subject activities table and enrol into it again
Select Cmp1 activity 1 from the subject activities table and enrol into it
[y] The "enrolled" column shows 1 for Lec1 activity 1 and 1 for Cmp1 activity 1 in both tables
Select Cmp1 activity 2 from the subject activities table and enrol into it
[?] The "enrolled" column shows 1 for Lec1 activity 1, shows 0 for Cmp1 activity 1, and shows 1 for Cmp1 activity 2
[?] Cmp1 activity 1 is removed from the "My Activities" table and Cmp1 activity 2 is added in its place
Select "31284 Web Services Development" from the subjects ComboBox
[y] The table of activities changes to show the activities for this subject (i.e. 1 Lec1 and 4 Cmp1 activities)
Enrol into Lec1 activity 1 and Cmp1 activity 1
[y] The "My Activities" table is updated to include these two activities, each with an enrolled column of 1
Click "Logout"
[y] The "Student" window is closed
Login to Bianca Sladen's account again.
[?] The "Student" window is shown, and the "My Activities" table shows that Bianca is still enrolled in the same 4 activities selected earlier, and the enrolled column shows 1 for each of those activities.
Click "Logout"
Login to Hugo Aitken's account
[y] The heading shows "Logged in as Hugo Aitken"
[y] The student number is shown as 49287512
[?] The attendance is shown as "Full Time"
[?] The scholarship is shown as "No"
[y] The "My Activities" table shows "Not enrolled in any activities"

Select Applications Programming from the subjects ComboBox
[y] It is shown that Lec1 activity 1 and Cmp1 activity 2 each already have one student enrolled
Enrol Hugo into the same two activities
[y] It is immediately shown that Lec1 activity 1 and Cmp1 activity 2 now have two students enrolled in each
Click "Logout"
Login to Jessica Sneddon's account
[?] The heading shows "Logged in as Jessica Sneddon", student number shows 23232323, attendance "Part Time", scholarship "Yes"
Select Applications Programming from the subjects ComboBox
[y] It is shown that Lec1 activity 1 and Cmp1 activity 2 each already have two students enrolled
Select Lec1 activity 1
[?] The "Enrol" button becomes Enabled
Enrol Jessica into Lec1 activity 1
[y] This activity is added to the "My Activities" table and the "enrolled" column shows 3. The subject activities list shows 3 students are enrolled into Lec1 activity 1 and 2 students are enrolled into Cmp1 activity 2
Select Cmp1 activity 2
[?] The "Enrol" button becomes disabled
Select Web Services Development from the subjects ComboBox
[y] The subject activities table shows Lec1 activity 1 and Cmp1 activity 1 each have 1 student enrolled
Click "Logout"
From the "University" window, select Bianca Sladen from the list and click "Remove Student"
[y] The student list in the "University" window is immediately updated to reflect that Bianca was removed
Login to Jessica Sneddon's account again
Select Applications Programming from the subjects ComboBox
[?] It is shown that Lec1 activity 1 now has 2 students enrolled and Cmp1 activity 2 now has 1 student enrolled
Select Cmp1 activity 2
[?] The "Enrol" button becomes enabled
Click "Enrol"
[?] The selected activity is added to the "My Activities" table and the enrolled column for this activity increases from 1 to 2 in both tables
Select Web Services Development from the subjects ComboBox
[?] It is shown that no student is enrolled into any activity in this subject

Use the Bank demo as a guide

The Bank demo is downloadable from UTSOnline -> Subject Documents -> Lecture 10 Code based on TableView. The ListView version is also downloadable as Lecture 9 Code. You need to study the Bank demo and keep it open as a reference while attempting the assignment. Observe the following parallels:

BankApplication shows shows the first window.	TimetableApplication shows the first window.
The Customer window displays a list of	The University window displays a list of

accounts. The model is the customer.	students. The model is the university.
Clicking the “Add Account” button opens the Add Account window. The model is the customer.	Clicking the “Add Student” button opens the Add Student window. The model is the university.
Clicking the “View Account” button opens the Account window. The model is the account.	Clicking the Login button opens the Student window. The model is the student.

A layout template

Every window has 3 sections arranged vertically in a VBox:

- A header section (this is an HBox)
- A content section (add whatever you want here)
- A footer section (this is an HBox)

For example:

```
<VBox styleClass=”root”>
     <HBox styleClass=”header”>...</HBox>
          <HBox styleClass=”content”>...</HBox>
     <HBox styleClass=”footer”>...</HBox>
</VBox>
```

Using styleClass=”header” means that later you will be able to apply CSS to make the header look the same colour on every window. If CSS is a “difficult bit” for you, save it until the end.

Once you get this template right, you can copy and paste it into all 3 FXML files as a starting point. (Note: the VBox above is the root element, so it should also have the xmlns:fx attribute and the fx:controller attribute.

Build the University window

- Write one line of code in TimetableApplication to show the /view/university.fxml view.
    - The model is a new university
    - The view is /view/university.fxml
    - The title is “University”
    - The stage is the primary stage
- Write a controller called UniversityController in the controller package which extends Controller<University>. Link the university.fxml file to it using fx:controller=”controller.UniversityController”. Run your application and fix any errors

- Expose a “university” property in the controller which represents the model.
  i.e. `public final University getUniversity() { return model; }`
- Show the image view and the heading text in the header section.
- Show the list view in the content section.
- Show the buttons in the footer section. Using FXML code, write disable=”true” for any button that should start off being disabled.
- When you make a ListView for this window, bind it to the list of students in the model. The binding expression is “${controller.university.students}”. To make it work, the controller **must** expose a “university” property (which you have already done), and the University **must** expose a “students” property which needs to be an ObservableList. Refer to the demo to see how this is done.
- So that you can see some students, it will be helpful to add some “dummy” data to the model. In your TimetableApplication, you wrote one line to show the first window, beginning with:

  ```
  ViewLoader.showStage(new University(), .....
  ```

  Change this to the following *test* code:

  ```
  University university = new University();
  university.addStudent(“12341234”, “John Smith”, “pt”, false);
  university.addStudent(“98989898”, “Susan Malone”, “ft”,
  true);
  ViewLoader.showStage(university, .....
  ```

  So whenever you launch the application, the university will already show two students. Don’t forget to remove this test code before your final version is submitted to PLATE!

Build the Add Student window

- Write onAction handler for the “Add Student” button so that when you click on this button, it opens the “Add Student” window:
  - The model is the university
  - The view is /view/add_student.fxml
  - The title is “Add Student”
  - The stage is a new stage
- Copy your VBox template into the /view/add_student.fxml file so that it has a header, content and footer section.
- Set the fx:controller attribute to point to controller.AddStudentController.
- Define an AddStudentController class in the controller package which extends Controller<University> (because the model for this view is the University).
- Expose a “university” property in the controller.
- Run and test the “Add Student” button to check that it works.
- Add a heading to the header section

- Add a GridPane to the content section showing all the labels and controls to add a student.
- Add the buttons to the footer section.
- Make the “Cancel” button close the window.
- Make the “Add” button add a new student and close the window. This involves the following steps:
    - Implement the getter patterns for each input: for the student number, student name, attendance and scholarship.
    - Implement an onAction handler for the “Add” button which calls the addStudent() method on the university. Refer to the Bank demo to see how this is done.
    - Because addStudent() can fail, you should put a try/catch block around your call to addStudent(). Leave the catch block empty for now.
- Test that your “Add” button works.

Build the Student window

- Define a change listener so that the “Remove Student” and “Login” buttons become enabled whenever a student is selected in the ListView. See the Bank demo for how to do this.
- Define an onAction handler for the “Login” button so that it opens the “Student” window:
    - The model is the selected student
    - The view is /view/student.fxml
    - The title is “Student”
    - The stage is a new stage
- Copy and paste your VBox template into the /view/student.fxml file, and using the same steps as usual, link up the FXML file to a controller for this window. You need to define this class so that it extends Controller with <Student> as the type parameter. In this controller, expose the model as a “student” property.
- Run and test the “Login” button to ensure that the new window appears.
- In preparation for displaying all of the data that is required for this view, you need to expose all of the model data as properties using the appropriate property patterns. Review the 4 property patterns now so that you understand what you have to do.
    - Activity properties:
        - The activity’s group, number, day, start, duration, room and capacity never change.
        - The enrolled count CAN change, although there is no “set” method. It changes via the enrol and withdraw methods.
        - You can skip the subject number. It is a “difficult bit”.
    - Student properties:
        - The student’s university remains the same, although it has mutable state.
        - The student number, name, attendance and scholarship never change.

        - The student's activities list is a fixed reference, although its contents can change (mutable state). Also, this list must be observable so that observers are notified of changes to the contents of the list.
    - University properties:
        - The students and subjects lists are fixed references, although their contents can change (mutable state). Also, these lists must be observable so that observers are notified of changes to the contents of the lists.
- Now you can bind the FXML views to the model properties. First, bind to the student's properties:
    - Bind the student number so it shows.
    - Bind the student name so it shows.
    - Skip the attendance and scholarship, they are difficult bits.
    - Bind the "My Activities" TableView to the student's activities list.
- The above bindings are the straightforward ones, because they are direct properties of the student, which is the primary model for this window. But this window also wants to view properties of the university. For this purpose, the student actually has a getUniversity() method and thus a "university" property. With this in mind:
    - Bind the ComboBox to the student's university's subjects.
- All of the bindings above can be done purely in FXML using binding expressions such as "${controller.student.name}" and so on. FXML bindings are "static" in the sense that the student name is **always** bound to "${controller.student.name}". But what about the second TableView? This binding would be "dynamic" in the sense that this TableView needs to be re-bound to a *different* list of activities each time you choose a subject from the ComboBox.
  An alternative to static FXML bindings is to set the items in Java using: `mytableview.setItems(...)` where `...` is the observable list that you want to display. You can call this method in Java whenever you want to change the list of items displayed. You should register a ChangeListener with the ComboBox so that whenever a different subject is selected, you call the setItems method to change the list that is displayed.
- To implement the enrol button, what you need is a reference to the student and the activity that student wants to enrol into. You already have a getStudent() method, so you just need a getter for the currently selected activity of the second TableView. This getter pattern is included in the lecture notes.
- To implement the withdraw button, it is similar except that you need another getter around your first TableView to get the currently selected activity from "My Activities". You can't use the same method name for these two getters, so name the two getters according to what TableView it is being selected from. The naming is up to you.

Second pass: 90%

After passing the following additional 19 verification points, you will receive a cumulative 75 potential marks on PLATE. Assuming you also met all of the coding requirements, including the correct use of exceptions, you should receive 15 marks after the due date giving you a total of 90%.

The additional verification points are highlighted in blue. Be careful when ticking off verification points involving the enabling and disabling of buttons because if your buttons don’t enable when they should, and if this prevents the user from clicking the button when they should be able to, this prevents you from getting marked for the following features. So don’t disable a button unless you also have correct code to enable that button when required.

Launch the application
[y] The UTS Logo is shown
[y] The "Timetable System" heading is shown
[y] "No students" is shown in the empty list
[y] "Add Student" appears and is enabled
[y] "Remove Student" appears and is disabled
[y] "Login" appears and is disabled
[y] The layout arranges the header, content and footer sections vertically
[y] All colours and fonts are correct
[y] The spacing and padding look like the screenshots
Click "Add Student"
[y] The "Add Student" window is shown
[y] The "Add new student" heading is shown
[y] The student number label and text field are shown
[y] The student name label and text field are shown
[y] The attendance label, "Full Time" radio button and "Part Time" radio button are shown
[y] The scholarship label and checkbox are shown
[y] The "Cancel" button is shown and enabled
[y] **The "Add" button is shown** and disabled
[y] The layout arranges the header, content and footer sections vertically
[y] The form is laid out in a GridPane
[y] All colours and fonts are correct including the error message
[y] The error message is included in the GridPane, centred horizontally spanning 2 columns
[?] The spacing and padding look like the screenshots
Click "Cancel"
[y] The "Add Student" window is closed
Click "Add Student" again
Enter student number 12345678
[y] The "Add" button remains disabled
Enter student name Bianca Sladen
[y] The "Add" button remains disabled
Select "Full Time"
[y] The "Add" button becomes enabled
Select the "Scholarship" checkbox
Delete the student number and the re-enter it
[y] The "Add" button becomes disabled and then re-enabled
Delete the student name and then re-enter it
[y] The "Add" button becomes disabled and then re-enabled
Click the "Add" button
[y] The "Add Student" window is closed

[y] "12345678 - Bianca Sladen" is shown in the student list in the university window
Click the "Add Student" button
Enter details "12345678", "Hugo Aitken", "Full Time" and no scholarship
Click "Add"
[y] The window does not close and the error "Student already exists" is shown
Change the student number to "49287512" and click "Add"
[y] The "Add Student" window is closed and "49287512 - Hugo Aitken" is immediately added to the student list in the university window
Add a student "23232323", "Jessica Sneddon", "Part Time", no scholarship
Add a student "11111111", "Dakota Cavill", "Full Time", scholarship
[y] These two students are immediately added to the student list in the university window
Select Bianca Sladen from the list
[y] At this moment, the "Remove Student" and "Login" buttons become enabled
Click "Login"
[y] The "Student" window is shown
[y] The "Logged in as Bianca Sladen" heading is shown
[y] The student number label and text (12345678) is shown
[y] The student attendance label and text ("Full Time") is shown
[y] The student scholarship label and text ("Yes") is shown
[y] **The "Withdraw" and "Enrol" buttons are both shown** and disabled. **The "Logout" button is shown** and enabled.
[y] The "My Activities" table shows column headings: Subject, Group, Activity, Day, Start, Duration, Room, Capacity, Enrolled.
[y] The "My Activities" table is empty and shows "Not enrolled in any activities"
[y] The subjects ComboBox is shown and it displays the subjects from the model (i.e. "48024 Applications Programming" and "31284 Web Services Development")
[y] The subject activities table is empty and shows column headings: Subject, Group, Activity, Day, Start, Duration, Room, Capacity, Enrolled.
[y] The layout arranges the header, content and footer sections vertically
[y] The "Logged in as ..." heading is in the top left, and the student details GridPane is in the top right
[y] The "My Activities" heading is aligned left, and the "Withdraw" button is aligned right
[y] The "Enrol into subject" heading and ComboBox are aligned left, and the "Enrol" button is aligned right
[y] All colours and fonts are correct
[?] The spacing and padding look like the screenshots
Select "48024 Applications Programming" from the subjects ComboBox
[?] The Subject column shows the correct subject number (and ONLY the subject number) of each activity in the selected subject
[y] The Group, Activity, Day, Start, Duration, Room, Capacity and Enrolled columns show the corresponding properties from the Activity model data for each activity in the subject
Select Lec1 activity 1 from the subject activities table
[y] The "Enrol" button becomes enabled
Click "Enrol" and make sure Lec1 activity 1 is still selected in the table
[?] The "Enrol" button becomes disabled
[y] The selected activity is immediately added to the student's activities and shows in the "My Activities" table
[y] The "Enrolled" column for the selected activity increases to 1 in both tables
Select Lec1 activity 1 from the "My Activities" table
[y] The "Withdraw" button becomes enabled
Click "Withdraw"

[y] The selected activity is removed from the "My Activities" table
[y] The "enrolled" column for that same activity is decreased to 0 in the subject activities table.
[y] The "Withdraw" button becomes disabled
Select Lec1 activity 1 from the subject activities table and enrol into it again
Select Cmp1 activity 1 from the subject activities table and enrol into it
[y] The "enrolled" column shows 1 for Lec1 activity 1 and 1 for Cmp1 activity 1 in both tables
Select Cmp1 activity 2 from the subject activities table and enrol into it
[?] The "enrolled" column shows 1 for Lec1 activity 1, shows 0 for Cmp1 activity 1, and shows 1 for Cmp1 activity 2
[?] Cmp1 activity 1 is removed from the "My Activities" table and Cmp1 activity 2 is added in its place
Select "31284 Web Services Development" from the subjects ComboBox
[y] The table of activities changes to show the activities for this subject (i.e. 1 Lec1 and 4 Cmp1 activities)
Enrol into Lec1 activity 1 and Cmp1 activity 1
[y] The "My Activities" table is updated to include these two activities, each with an enrolled column of 1
Click "Logout"
[y] The "Student" window is closed
Login to Bianca Sladen's account again.
[?] The "Student" window is shown, and the "My Activities" table shows that Bianca is still enrolled in the same 4 activities selected earlier, and the enrolled column shows 1 for each of those activities.
Click "Logout"
Login to Hugo Aitken's account
[y] The heading shows "Logged in as Hugo Aitken"
[y] The student number is shown as 49287512
[y] The attendance is shown as "Full Time"
[y] The scholarship is shown as "No"
[y] The "My Activities" table shows "Not enrolled in any activities"
Select Applications Programming from the subjects ComboBox
[y] It is shown that Lec1 activity 1 and Cmp1 activity 2 each already have one student enrolled
Enrol Hugo into the same two activities
[y] It is immediately shown that Lec1 activity 1 and Cmp1 activity 2 now have two students enrolled in each
Click "Logout"
Login to Jessica Sneddon's account
[y] The heading shows "Logged in as Jessica Sneddon", student number shows 23232323, attendance "Part Time", scholarship "Yes"
Select Applications Programming from the subjects ComboBox
[y] It is shown that Lec1 activity 1 and Cmp1 activity 2 each already have two students enrolled
Select Lec1 activity 1
[y] The "Enrol" button becomes Enabled
Enrol Jessica into Lec1 activity 1
[y] This activity is added to the "My Activities" table and the "enrolled" column shows 3. The subject activities list shows 3 students are enrolled into Lec1 activity 1 and 2 students are enrolled into Cmp1 activity 2
Select Cmp1 activity 2
[y] The "Enrol" button becomes disabled
Select Web Services Development from the subjects ComboBox
[y] The subject activities table shows Lec1 activity 1 and Cmp1 activity 1 each have 1 student enrolled
Click "Logout"

From the "University" window, select Bianca Sladen from the list and click "Remove Student"
[y] The student list in the "University" window is immediately updated to reflect that Bianca was removed
Login to Jessica Sneddon's account again
Select Applications Programming from the subjects ComboBox
[?] It is shown that Lec1 activity 1 now has 2 students enrolled and Cmp1 activity 2 now has 1 student enrolled
Select Cmp1 activity 2
[?] The "Enrol" button becomes enabled
Click "Enrol"
[?] The selected activity is added to the "My Activities" table and the enrolled column for this activity increases from 1 to 2 in both tables
Select Web Services Development from the subjects ComboBox
[?] It is shown that no student is enrolled into any activity in this subject

General tips for the second pass

- You may get NullPointerExceptions when handling the attendance toggle group. Some help on this is provided in the FAQ.
- Refer to the Bank demo, particularly the week 10 lecture code, for a more complex example of how to enable and disable buttons based on the status of multiple form inputs.
- You need to register ChangeListeners on EVERY control that could potentially change the disable property of a button. For example, on the “Add Student” window, you need to set a ChangeListener on the name text field, AND the number text field, AND the attendance toggle group because changing any of those could potentially change the status of the button. But you do not need to set a ChangeListener on the scholarship checkbox because changing that will have no effect on the status of the button.
- For the exception, you don’t need to create a custom exception. A generic exception is fine. However, the exception should include within it an error message, as shown in the lecture slides, such that e.getMessage() can be used to obtain that error message.
- The error message in the “Add Student” window is just a Text node. It is always visible, but starts off with a text property of just “”, the empty string. When you catch the exception, you set the text property of this Text node to the message contained within the exception.